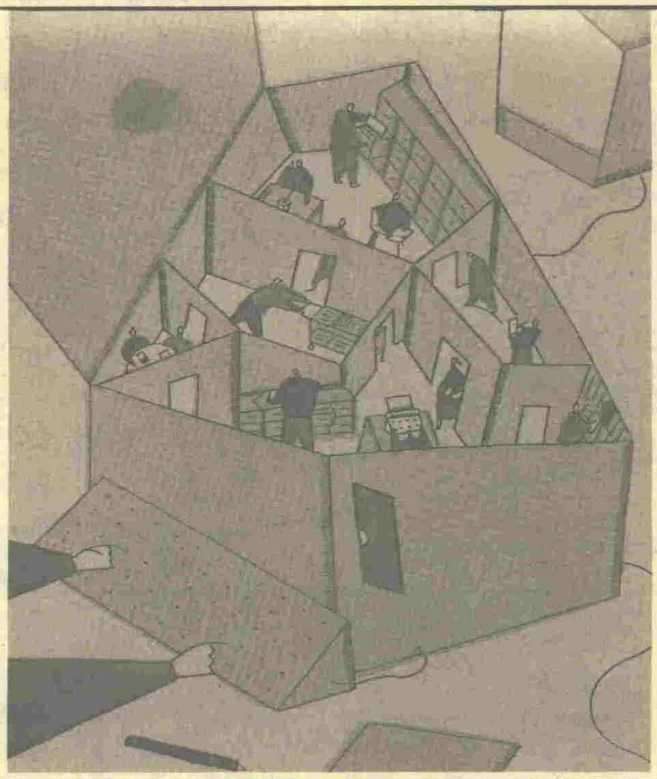
OOP
ความหวังของการเก็บของเก่ามาใช้ใหม่



กองบรรณาธิการ

ในช่วงเวลาเจ็ดแปดปีที่ผ่านมา มีการกล่าวถึงความสูญเสียของการเขียนซอฟต์แวร์แล้วไม่ได้ใช้อยู่มาก มีซอฟต์แวร์จำนวนมากเมื่อพัฒนาขึ้นมาแล้วประกาศขายแต่ไม่ประสบผลสำเร็จในเรื่องการขาย บางโปรแกรมได้ลงทุนไปเป็นจำนวนมหาศาล มีการพัฒนาเทคนิคต่าง ๆ ขึ้นมาใช้ แต่ครั้นเทคโนโลยีทางฮาร์ดแวร์เปลี่ยนไป ซอฟต์แวร์เหล่านั้นก็ตายหายจากไป พวกเราคงจะเห็นแล้วว่าในยุคไมโครคอมพิวเตอร์ 8 บิต (ซีพี/เอ็ม) ก็มีซอฟต์แวร์ที่พัฒนามาใช้กันมาก ครั้นมายุค 16 บิต ซอฟต์แวร์ของ 8 บิตก็เลิกใช้ และไม่สามารถนำกลับมาใช้ได้ใหม่ เรื่องดังกล่าวนี้นับเป็นความสูญเสียอย่างมหาศาลในวงการซอฟต์แวร์คอมพิวเตอร์

หลังจากปี ค.ศ. 1981 เป็นต้นมา นักคอมพิวเตอร์เป็นจำนวนมากเห็นปัญหาดังกล่าวและพยายามพัฒนาซอฟต์แวร์ในลักษณะให้สามารถนำกลับมาใช้ใหม่ได้ การพัฒนาหลักการจึงก้าวหน้าขึ้นเป็นลำดับ และนี่เองเป็นจุดเริ่มต้นของแนวความคิดที่เรียกว่า ออปเจ็คโอเรียนเต็ดโปรแกรม (Object Oriented Programming) หรือเรียกย่อ ๆ ว่า โอโอพี

โอโอพี คืออะไร

ลองนึกถึงการดำเนินกิจกรรมต่าง ๆ ในชีวิตประจำวัน แล้วเปรียบเทียบกับการทำงานของโปรแกรมคอมพิวเตอร์ สิ่งที่พบเห็นจะมีหลายอย่างที่มีลักษณะที่ใกล้เคียงกัน และสามารถนำมาใช้ได้ในหลักการสมัยใหม่ของโปรแกรมคอมพิวเตอร์

ต้นฉบับไม่มีหน้า 239-240

No This Page 239-240 In Original

สมมติว่าคุณต้องการส่งช่อดอกไม้ไปให้คนที่เป็นสุดที่รัก เนื่องในโอกาสวันวาเลนไทน์ แต่เจ้าหล่อนอยู่ห่างจากคุณเป็นระยะทางหลายร้อยกิโลเมตร และคุณก็ไม่มีโอกาสเดินทางไปมอบให้ด้วยตนเองได้ งานง่าย ๆ แค่นี้คุณย่อมต้องทำได้ คุณเริ่มจากการที่ได้เดินทางไปยังร้านขายดอกไม้ที่อยู่ใกล้ที่สุด อธิบายกับเจ้าของร้านว่าคุณต้องการดอกไม้ชนิดอะไร จำนวนเท่าไร ส่งไปให้ใครที่ไหน เท่านี้คุณก็ทำงานสำคัญยิ่งของชีวิตคุณได้สำเร็จ

จากตัวอย่างที่ยกมานี้ลองพิจารณาดูว่าการดำเนินการของร้านดอกไม้ทำอะไรบ้าง ร้านดอกไม้ที่ได้รับคำสั่งจากคุณ ก็ส่งใบสั่งซื้อไปยังร้านค้าดอกไม้ที่อยู่ในเมืองที่แฟนคุณอยู่ ร้านค้าดอกไม้จะจัดเตรียมดอกไม้ตามที่ระบุ ซึ่งก็อาจจะต้องมีการไปติดต่อกับร้านขายส่งดอกไม้ หรือสั่งจองไว้ และถ้าติดตามต่อไปก็จะพบว่าร้านขายส่งดอกไม้ ก็มีการติดต่อกับชาวสวนดอกไม้ที่เป็นผู้ปลูก และมีความสัมพันธ์ต่อออกไปเรื่อย ๆ

ความจริงแล้วถ้าพิจารณาถึงวิธีการสั่งดอกไม้ ก็ไม่สนใจว่ากรรมวิธีการได้ดอกไม้มานั้นจะเป็นอย่างไร ขอให้ดอกไม้ไปถึงปลายทางเท่านั้นก็พอแล้ว ในชีวิตจริงจึงเป็นการกระจายอำนาจหน้าที่ออกไปให้ผู้รับผิดชอบต่อ ๆ กัน แต่ในทางคอมพิวเตอร์เราเรียกว่า "abstraction" หรือ "information hiding" กล่าวคือ กรรมวิธีหรือข้อมูลและข่าวสารภายใน เป็นสิ่งที่เราไม่สนใจที่จะรู้ แต่ขอให้ได้เอาต์พุตหรือผลลัพธ์ตามที่ต้องการอันเป็นการใช้ได้

โมเดลของโอโอพี จึงเป็นการแก้ปัญหาต่าง ๆ ในลักษณะที่ได้ยกตัวอย่างมาแล้วนี้ ซึ่งเห็นได้ว่า ผู้ที่ไม่ได้รับการฝึกฝนทางด้านคอมพิวเตอร์ และไม่เคยใช้คอมพิวเตอร์มาก่อนจะมีความรู้สึกว่าวิธีการของโอโอพีจะเป็นธรรมชาติใกล้เคียงกับชีวิตประจำวันมากกว่า แต่ผู้ที่เคยเขียนโปรแกรมคอมพิวเตอร์หรือผ่านการศึกษาในวิทยาลัย หรือมหาวิทยามาแล้วจะรู้สึกว่า โอโอพีเป็นเรื่องที่แปลกและขัดกับความรู้สึกในเรื่องโปรแกรม

เช่น "7" ถือได้ว่าเป็นออปเจ็ค และ "+" คือสิ่งที่ต้องการบอกให้สำหรับการบวก ซึ่งดูเป็นเรื่องแปลก เพราะจะมองสิ่งต่าง ๆ ในโปรแกรมเป็นออปเจ็ค เช่น จากตัวอย่างในเรื่องการส่งดอกไม้ "คุณ" เป็นออปเจ็ค "ร้านขายดอกไม้" ก็เป็นออปเจ็คมีการส่งข้อความระหว่างออปเจ็ค ออปเจ็คเมื่อได้รับข้อความก็จะเอ็กซีคิวต์ว่าคืออะไร ให้ทำอะไร และจะส่งข้อความต่อไปที่ใด อย่างไร กล่าวคือออปเจ็คจะตอบสนองต่อข้อความที่มันได้รับ

ความคิดของโอโอพีกลายมาเป็น Smalltalk

ในปี ค.ศ. 1981 วารสารไบต์ฉบับเดือนสิงหาคมได้จัดทำบทความพิเศษทั้งฉบับในเรื่องของภาษา smalltalk ซึ่งในขณะนั้นภาษา smalltalk ยังเป็นเพียงโปรโตไทป์ ยังไม่มีขายในเชิงการค้า ความคิดทางด้านออปเจ็คจึงเป็นความคิดที่มีจุดเริ่มต้นและแพร่ขยายเข้าไปในภาษาคอมพิวเตอร์อื่น ๆ ตั้งแต่บัดนั้น

ปัจจุบันความคิดของโอโอพีได้รับการนำมาประยุกต์ใช้ในภาษาต่าง ๆ มาก และเชื่อว่า ความคิดของออปเจ็คโอเรียนเต็ดโปรแกรมจะได้รับความนิยมกันในอนาคต เหมาะทำให้โปรแกรมมีประสิทธิภาพและสามารถนำมาใช้ในการแก้ปัญหาที่ซับซ้อนให้ดูง่ายขึ้น งานพัฒนาโปรแกรมขนาดใหญ่และซับซ้อนจะทำได้สะดวกขึ้น และพัฒนาเพิ่มต่อได้ง่าย

โอโอพี ได้รับการพัฒนาต่อมาในเรื่องของแนวคิดและหลักการ ผู้ผลิตซอฟต์แวร์ภาษายุคใหม่ได้นำความคิดนี้ผนวกเข้าด้วยเสมอ และมีแนวโน้มที่จะเป็นมาตรฐานของการใช้กันอย่างแพร่หลายต่อไป

พื้นฐานของโอโอพี

บทความนี้จะเน้นการสำรวจหลักการพื้นฐานและความเป็นไปของโอโอพี เพื่อกระตุ้นให้นักคอมพิวเตอร์ได้สนใจในการที่จะศึกษาต่อ และหาทางนำไปใช้ต่อไป สิ่งที่เป็นหลักการพื้นฐานของโอโอพี ในแก่นหลักคือ ออปเจ็ค คลาส และอินเฮอริแตนซ์ (object, classes, inheritance) ซึ่งจะต้องอธิบายหลักการนี้ให้เข้าใจเสียก่อน

ออปเจ็คคืออะไร

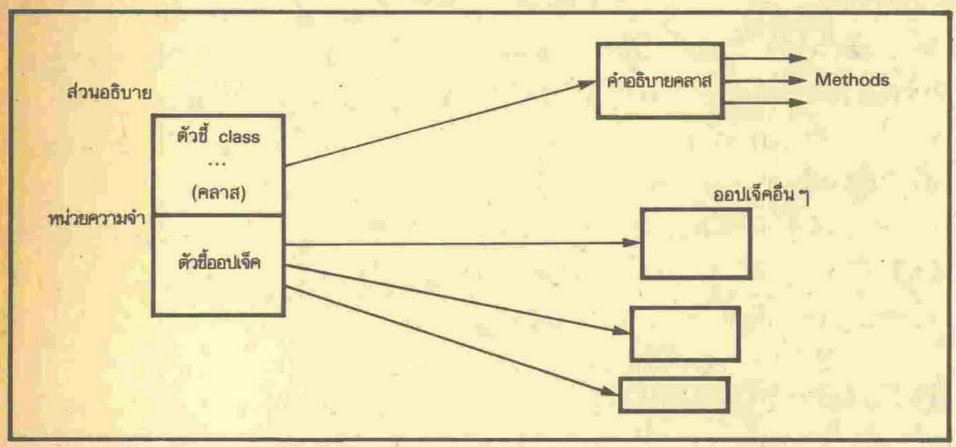
ความจริงแล้ว แนวความคิดในเรื่องออปเจ็คมีมานานแล้ว ในเรื่องของฐานข้อมูลก็มีการมองเอ็นติตี้ (entity) ของข้อมูล ซึ่งเปรียบเสมือนเป็นออปเจ็ค เอ็นติตี้ของข้อมูลมีความสัมพันธ์กัน เช่น ลูกค้ากับสินค้า ทั้งลูกค้าและสินค้าเป็นเอ็นติตี้ ในภาษาโลโกมีการมองสิ่งต่าง ๆ เป็นออปเจ็คอยู่แล้ว เช่น หุ่นยนต์ประกอบด้วยท่อนหัว ท่อนลำตัว แขน ขา เป็นต้น เมื่อนำแต่ละส่วนมาประกอบกันก็มีการเชื่อมต่อสัญญาณบางสิ่งบางอย่างเข้าด้วยกัน โปรแกรมตามแนวความคิดของภาษาโลโกจึงจัดได้ว่ามีรากฐานของความคิดทางด้านโอโอพี

ออปเจ็คมีคุณสมบัติหลายอย่างทุก ๆ ออปเจ็คประกอบด้วยส่วนการเก็บข้อมูลในหน่วยความจำ ที่สามารถเรียกเข้าหาได้ หรือมองเห็นเสมือนเป็นกลุ่มก้อน ออปเจ็คแต่ละตัวไม่สามารถเข้าไปดัดแปลงแก้ไขข้อมูลในหน่วยความจำของออปเจ็คอื่น ดังนั้น ในระบบหนึ่งจะประกอบด้วยหลาย ๆ ออปเจ็ค ในหน่วยความจำของออปเจ็คหนึ่งก็เป็นที่อยู่ของออปเจ็คอื่นได้ การแทน

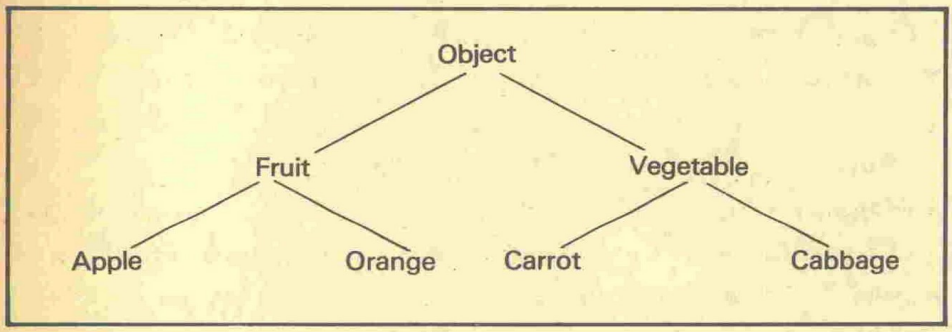
ออปเจ็คในแต่ละภาษาจึงใช้หลักการแทนด้วยโครงสร้างข้อมูล ซึ่งถ้ามองในระดับโปรแกรม เช่น ภาษาปาสคาล ก็มองเห็นออปเจ็คเสมือนเป็นข้อมูลแบบเรคอร์ด และถ้าเป็นซี ก็มองเห็นเสมือนเป็นข้อมูลโครงสร้าง (structure) โครงสร้างของออปเจ็คเขียนเป็นไดอะแกรมดังรูปที่ 1

การดำเนินการของโปรแกรมเป็นการสร้างการส่งผ่าน message โดย message หรือข้อความนี้ เป็นสิ่งที่ออปเจ็คต้องการสำหรับการทำงานบางอย่าง และก็ยังต้องการอาร์กิวเมนต์ของการเอ็กซีคิวต์การทำงาน แต่การส่งข้อความนี้ถ้าเป็นแบบภาษาแบบโพรซีเดอร์ เช่น ปาสคาลก็จะใช้การเรียกผ่านโปรแกรมย่อย ซึ่งในขณะโปรแกรมย่อยทำงาน โปรแกรมที่เรียกมาจะหยุดและรอจบรีเทอร์นค่าหรือการทำงานให้แล้วเริ่มทำงานต่อ แต่สำหรับออปเจ็คในโอโอพี มีการทำงานที่ไดนามิกส์กว่านี้อีก

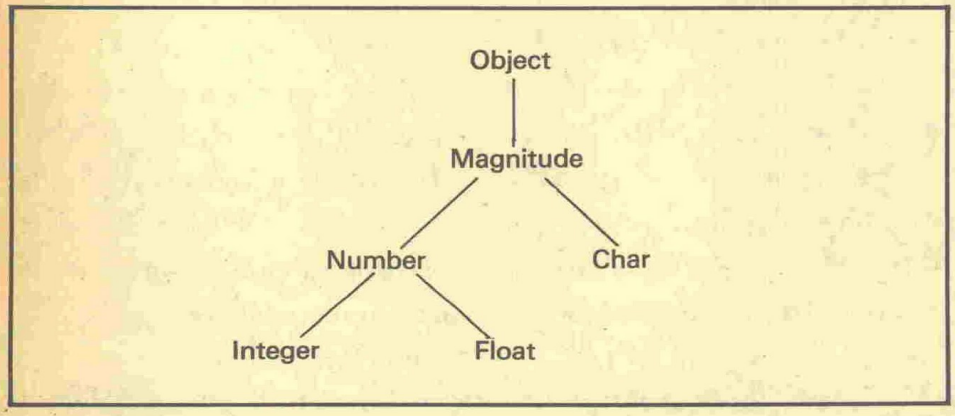
ในโลกของความเป็นจริง วัตถุหรือออปเจ็คก็มีลักษณะเหมือนกับออปเจ็คในโอโอพี เช่น ในตะกร้าแอปเปิ้ลจะเห็นว่า แอปเปิ้ลแต่ละลูกก็ยังแตกต่างกัน แต่แอปเปิ้ลทุกลูกก็มีลักษณะเหมือนกันในคุณสมบัติบางอย่าง เช่น กลิ่น รส และสามารถนำมาทำอาหารอย่างอื่น เช่น พายแอปเปิ้ล เราจะมองแอปเปิ้ลทั้งหมดที่มีคุณสมบัติเหมือนกันนี้ในรูปแบบการจัดจำแนกออกจากวัตถุอื่นที่เรียกว่า คลาส (CLASS)



รูปที่ 1 โครงสร้างออปเจ็คพื้นฐาน



รูปที่ 2 โครงสร้างลำดับชั้นของคลาสและซับคลาส



รูปที่ 3 โครงสร้างคลาสของตัวเลข

คลาส

คลาสเป็นวิธีการแยกออปเจ็คออกจากกัน เช่น แอปเปิ้ลกับส้ม จะอยู่คนละคลาสแต่ทั้งหมดนี้รวมกันเรียกว่า ผลไม้ ผลไม้จึงเป็น superclass ของแอปเปิ้ล, ส้มหรืออื่น ๆ ในกรณีนี้เราอาจเรียกได้ว่า ส้มเป็น subclass ของผลไม้

การจัดแบ่งคลาสจึงเป็นโครงสร้างลำดับชั้น ลองพิจารณาโครงสร้างลำดับชั้นดังรูปที่ 2 ซึ่งเป็นการแบ่งให้ของสิ่งต่าง ๆ ออกเป็นคลาสย่อย ๆ

ในเรื่องของโอโอพีก็เช่นเดียวกัน ทุก ๆ ออปเจ็คจะต้องมีคลาส และบางออปเจ็คก็มี subclass หรือ superclass ซึ่งมีโครงสร้างในลักษณะคล้ายกับรูปที่ 2 ในบางภาษาเช่น smalltalk กำหนดชื่อ class ด้วยตัวอักษรตัวแรกเป็นตัวอักษรใหญ่ เช่น เลข 7 อยู่ในคลาส Integer 8 ก็อยู่ในคลาส Integer ด้วย 7 และ 8 เป็นออปเจ็คที่ต่างกัน ถ้าหากให้ออปเจ็ค 7 และ 8 ตอบสนองต่อ message + โดยมี integer argument สำหรับการบวก เป็นต้น

สังเกตว่า Integer เป็น subclass ของ Number Float ก็เป็น subclass ของ Number โครงสร้างคลาส ตัวเลขเขียนได้ดังรูปที่ 3

การที่ออปเจ็คมีการตอบสนองต่อ message ตามกรรมวิธีของ message นี้ เราเรียกว่า method เช่น ออปเจ็คในคลาส Integer method ซึ่งเกี่ยวกับการตอบสนองของ message " + "

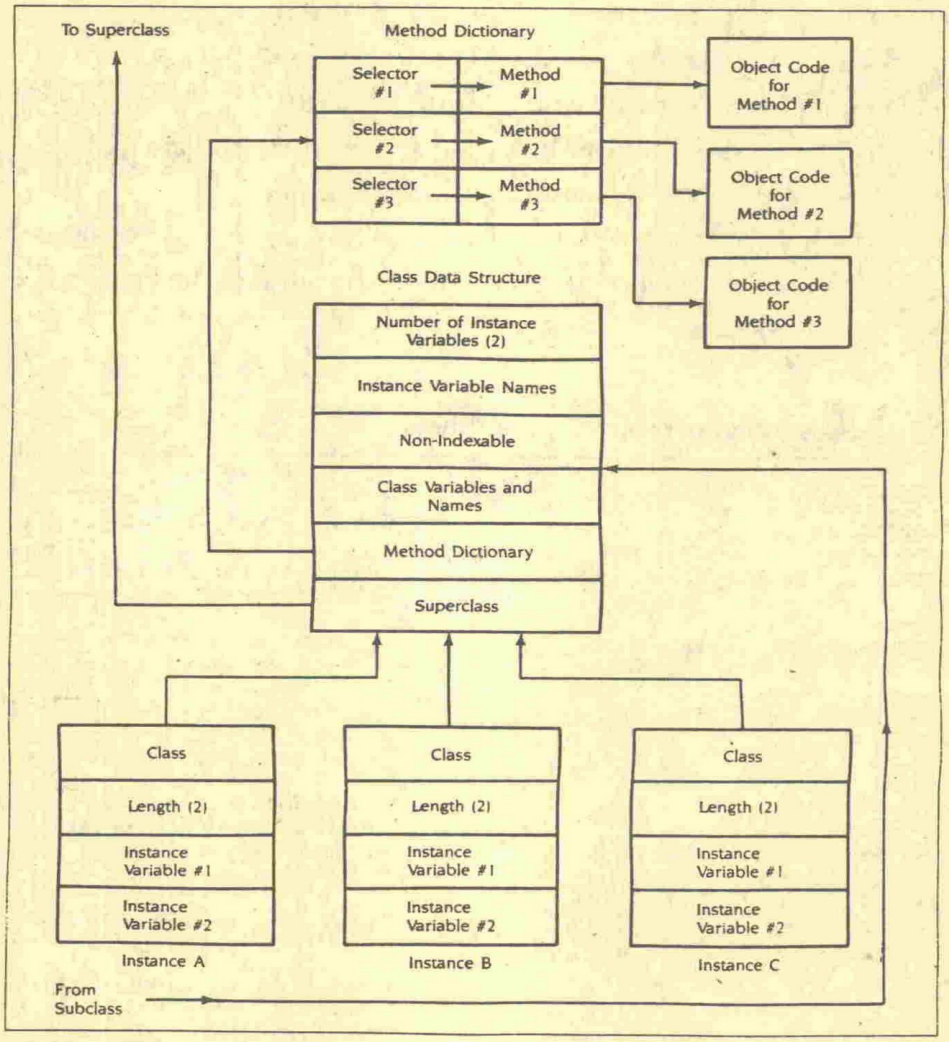
กลุ่มของ message ที่เกี่ยวกับการทำงานในคลาสเดียวกันเราเรียกว่า protocol เช่น คลาส Integer ประกอบด้วยโปรโตคอลของ message + − * / เป็นต้น

อินเฮริแตนซ์ (Inheritance)

หากออปเจ็คอยู่ในคลาสที่กำหนด ถ้าเรารู้ method ที่เกี่ยวข้องกับคลาสนั้น ก็สามารถใช้งานออปเจ็คนั้นได้ และหากเราพิจารณาในส่วนของ method นั้น มันจะเกี่ยวข้องกับ superclass หรือ subclass อย่างไร

ลองพิจารณาจากตัวอย่างต่อไปนี้ อาจทำให้เข้าใจในความหมายของคำว่าอินเฮริแตนซ์ได้ดีขึ้น สมมติมี message "exp" หมายความว่า หากออปเจ็ค 2 ตอบสนองต่อ message "exp" หรือ 2 exp หมายถึง e^2 ซึ่งจะได้ผลลัพธ์ว่ายกกำลังสองของค่า e (e = 2.71828...) ซึ่งจะได้ผลลัพธ์ประมาณ 7.38906

ครานี้ถ้าหากดูระบบที่เกี่ยวข้องออปเจ็ค 2 อยู่ในคลาส integer แต่ Integer เป็นคลาสที่ไม่มี method สำหรับ message "exp" ซึ่งคอมพิวเตอร์พยายามจะหา method สำหรับ message "exp" จากโปรโตคอลของคลาส Integer แต่หาไม่พบ ดังนั้น ระบบคอมพิวเตอร์ต้องไปดูใน superclass คือ Number ปรากฏว่าโปรโตคอลของ Number สามารถทำการประมวลผลได้ เราจะกล่าวได้ว่า method สำหรับ exp นั้น มีอินเฮริแตนซ์กับคลาส Integer มาจาก คลาส Number



รูปที่ 4 โครงสร้างข้อมูลของโอโอพี

โครงสร้างความสัมพันธ์ของข้อมูลภายในออปเจ็ค

จากรูปที่ 1 จะเห็นความสัมพันธ์ของออปเจ็คกับ method และ class แต่ละออปเจ็คจะมีโครงสร้างอยู่ในหน่วยความจำในลักษณะข้อมูลภายใน และเชื่อมกันระหว่าง class ดังแสดงในรูปที่ 4

โครงสร้างภาษาแบบโพรซีเดอร์เปรียบเทียบกับ OOP

ในภาษาบางภาษา เช่น ปาสคาล เราของเป็นโพรซีเดอร์หรือฟังก์ชัน เช่น การเขียน sqrt (x) เป็นโพรซีเดอร์หรือฟังก์ชันของ sqrt โดยมี x เป็นอาร์กิวเมนต์ ซึ่ง x จะเป็นพารามิเตอร์ที่มีการส่งค่าไปให้ แต่สำหรับภาษาที่เป็นแบบโอโอพีจะเขียนอีกรูปแบบเป็น

X : sqrt

ในกรณีนี้ x เสมือนเป็นตัวรับและตอบสนองต่อ message : sqrt

7 + 17 : sqrt

จะมีความหมายเป็น 7 + (17 sqrt) ไม่ใช่ (7+17) sqrt

เพื่อให้เห็นรูปโครงของการเขียนโปรแกรมในแบบโอโอพีกับแบบของปาสคาลเปรียบเทียบกัน ในที่นี้จะนำตัวอย่างโปรแกรมที่มีผู้เขียนเสร็จแล้ว คือ โปรแกรมหอคอยแห่งเมืองฮานอย ซึ่งเป็นโปรแกรมปัญหาที่รู้จักกันมาตั้งแต่ปี ค.ศ. 1960 โดยมีเรื่องเล่าว่าพระสงฆ์แห่งเมืองฮานอยต้องการย้ายแผ่นจานที่เรียงซ้อน ๆ กันบนเสาหลักจำนวน 64 ชิ้น ออกไปตั้งไว้อีกที่หนึ่ง แผ่นจานนี้มีขนาดเรียงกันจากฐานไปหายอดคือใหญ่ไปเล็ก ในการย้ายจะมีเสาหลักอีก 2 ต้น รวมเป็นสามหลัก ซึ่งการถอดจานออกจะต้องนำไปใส่ยังหลักใดก็ได้ แต่ทุกขณะของแต่ละเสาแผ่นจานที่วางซ้อนต้องให้แผ่นเล็กกว่าอยู่บน

ปรากฏว่าปัญหานี้เป็นปัญหาที่นักคอมพิวเตอร์นำมาใช้เป็นตัวอย่างของการแก้ปัญหาแบบรีเคอร์ซีฟกัน โดยตั้งกฎเกณฑ์ง่าย ๆ โดยสมมติว่ามีหลัก 3 หลักคือ A, B, C และต้องการย้ายแผ่นจาน n แผ่นจากเสาหลัก A ไปยังเสาหลัก C เราอาศัยกฎเกณฑ์ ดังนี้

1. Move n-1 disks from A to B
2. Move 1 disk from A to C
3. Move n-1 disk from B to C

ในกรณีนี้ถ้า n = 1 ก็เป็นการทำที่ง่ายมาก และเมื่อรู้ค่าของ n = 1 ก็ขยาย n = 2...n ตามที่ต้องการได้

โปรแกรมที่เขียนด้วยภาษาปาสคาลและ Smalltalk เพื่อเปรียบเทียบกันแสดงดังรูปที่ 5

แนวทางพัฒนาพยายามสร้างเข้าสู่ซอฟต์แวร์ไอซี

หากพิจารณาการทำงานของวงจรคอมพิวเตอร์ จะเห็นว่าประกอบด้วย

รูปที่ 5 โปรแกรมแบบโพรซีเดอร์และแบบออปเจ็คโอเรียนเต็ดโปรแกรม

```
program hanoi(input, output);
  var total : integer;
  procedure movetower (Height, Frompin,
            Topin, Usingpin : integer);

    procedure movedisk ( Frompin,
                         Topin : integer);
    begin
      writeln( Frompin, '->', Topin)
    end; { movedisk }

  begin { movetower }
    if Height > 0 then
      begin
        movetower(Height-1,  Frompin,
                  Usingpin, Topin);
        movedisk( Frompin, Topin);
        movetower(Height-1, Usingpin,
                  Topin, Frompin)
      end
  end; { movetower }

begin { hanoi }
     read(total);
     movetower(total,1,3,2)
end.  { hanoi }
```

```
moveTower: height from: fromPin to: toPin
                              using: usingPin
"Recursive procedure to move the disk at a
 height from one pin to another using a
 third pin"
   (height > 0) ifTrue: [
     self moveTower: (height - 1) from:
        fromPin to: usingPin using: toPin.
     self moveDisk: fromPin to: toPin.
     self moveTower: (height - 1 ) from:
        usingPin to: toPin using: fromPin]

"This comment gives an example of how to
 run this program.  Select the following
 text and choose 'do it' from the
 middle-button menu.

(Object new) moveTower: 3 from: 1 to:
                              3 using: 2 "

moveDisk: fromPin to: toPin

"Move disk from a pin to another pin.
 Print results in the transcript window"
   Transcript cr.
   Transcript show: (fromPin
    printString, '->', toPin printString)
```

ชิพหรือตัวไอซีต่าง ๆ มาประกอบกัน ไอซีแต่ละตัวก็มีขาที่รับสัญญาณอินพุต หรือส่งสัญญาณเอาต์พุตให้กัน การทำงานของระบบจึงประกอบด้วยไอซีต่าง ๆ ที่เป็นหน่วยย่อย ๆ ของวงจรทำงานร่วมกัน ผู้ออกแบบระบบเพียงนำเอาไอซีมาประกอบเข้าให้ถูกต้องตามการออกแบบ โดยอาจไม่ต้องรู้รายละเอียดการทำงานภายในของไอซีแต่ละเบอร์เลย

ออปเจ็คโอเรียนเต็ดโปรแกรมหรือโอโอพี ก็เดินเข้าหาแนวความคิดนี้ กล่าวคือ มองส่วนของโปรแกรมเป็นออปเจ็คเสมือนเป็นชิพของไอซีที่เชื่อมต่อกัน มีการทำงานตามสเปคของแต่ละออปเจ็ค เพื่อส่งสัญญาณเข้าออกระหว่างออปเจ็คและให้แต่ละออปเจ็คทำงาน (responce) ตาม message โดยแต่ละออปเจ็คจะทำงานตามกรรมวิธี (method) การเชื่อมโยงก็จะใช้โปรโตคอล ทำให้ผู้ออกแบบซอฟต์แวร์สามารถนำเอาออปเจ็คต่าง ๆ มาเชื่อมโยงต่อกันได้

ความคาดหวังต่อไปคือ หากเรามีซอฟต์แวร์ขายเป็นเสมือนเบอร์มาตรฐานของไอซี เช่น 7400 คือ Nand gate ผู้ใช้ก็เพียงเลือกออปเจ็คต่าง ๆ ที่กำหนดตามมาตรฐานเหล่านั้นออกมาเชื่อมโยงกันเข้าเพื่อสร้างระบบซอฟต์แวร์ใหม่ ด้วยระบบออปเจ็คมาตรฐานนี้ก็ได้เริ่มมีผู้สร้างไว้ให้บ้างแล้ว เช่น ในเครื่องแมคอินทอช มีการกำหนดการทำงานของ Window และ icon เพื่อให้ผู้ใช้นำไปสร้างซอฟต์แวร์ของตนเอง เสมือนการออกแบบวงจรทางฮาร์ดแวร์

ด้วยวิธีการออกแบบซอฟตแวร์เสมือนการออกแบบวงจร ทำให้การขยายระบบงานหรือการตรวจสอบ การดูแลรักษา ตลอดจนออปเจ็คเหล่านั้นจะนำมาใช้ได้กับเครื่องทุกเครื่อง ถ้าเครื่องคอมพิวเตอร์เหล่านั้นสนับสนุนซอฟต์แวร์ไอซีตามเบอร์มาตรฐาน

แนวความคิดนี้จึงเป็นความหวังของการใช้โอโอพี เพราะการเชื่อมโยงภายในออปเจ็คทำอย่างอัตโนมัติ จึงทำให้ผู้ออกแบบซอฟต์แวร์ออกแบบโปรแกรมได้ง่าย นับว่าเป็นข้อดีเด่นสำหรับความคิดนี้ ที่จะทำให้ซอฟต์แวร์นำกลับมาใช้ได้อีก

จุดอ่อนของโอโอพี

สืบเนื่องจากการเชื่อมโยงโครงสร้างออปเจ็คภายในเป็นแบบไดนามิกส์ ดังนั้น จึงต้องอาศัยเวลาในการทำงาน การส่งผ่าน message จะใช้เวลามากกว่าการเรียกโปรแกรมย่อยในแบบเก่า มีผู้ทำการคำนวณเวลาที่ใช้เปรียบเทียบกัน ปรากฏว่า วิธีการของโอโอพีใช้เวลามากเป็น 1.75 เท่าของการเรียกโปรแกรมย่อยในแบบเก่า อย่างไรก็ตามหากโปรแกรมยุ่งยากและมีการจัดการแบบออปเจ็คที่ดีแล้ว ก็จะทำให้ความเร็วโดยรวมของแบบโอโอพีดีขึ้น

ข้อเสียอีกประการหนึ่ง คือ การประยุกต์ด้วยโอโอพียังมีความยุ่งยากกว่าแบบโพรซีเดอร์ เพราะโครงสร้างความคุ้นเคยของแบบลำดับของนักโปรแกรมในขณะนี้ผูกพันธ์กับแบบเดิม และแบบโอโอพีก็เกี่ยวข้องกับ abstract data structure อยู่มาก ผู้ที่จะเขียนโปรแกรมแบบโอโอพีต้องทำความเข้าใจในเรื่อง abstract เหล่านี้ ก่อนจึงจะเขียนได้ดี

ภาษาที่รองรับ Object Oriented Programming

ภาษาคอมพิวเตอร์รุ่นใหม่ มักจะมีการเสริมให้สามารถใช้หลักการของการเขียนโปรแกรมแบบโอโอพีได้ ในที่นี้ขอเสนอแนะชื่อภาษาที่พอจะหามาทดลองใช้ได้

SmallTalk ภาษานี้เป็นต้นตำรับของโอโอพี SmallTalk ได้รับการพัฒนาโดยบริษัท Xerox ในราวประมาณปี ค.ศ. 1975

Actor ภาษานี้เป็นอินเตอร์เพรตเตอร์ที่ยังคงใช้การควบคุมโปรแกรมแบบ procedural ภาษา Actor สนับสนุนหลักการใช้ message และ method โดยมีไลบรารี่ภายในสำหรับการสร้างคลาส

C++ พัฒนาโดย Bjarne Strousetrup ที่เบลแลปของบริษัท AT&T โครงสร้างของภาษาเป็นการขยายจากภาษาซีเดิมออกไป การเขียนโปรแกรมต้องมีการกำหนดคลาส อย่างไรก็ตาม C++ ก็ยังมีข้อแตกต่างจากตัวจริงของโอโอพีอยู่ในรายละเอียด

เทอร์โบปาสคาล เวอร์ชัน 5.5

บอร์แลนด์เห็นความสำคัญของโอโอพีจึงเพิ่มขีดความสามารถของหลักการทางด้านออปเจ็คและคลาสเข้าไป เพื่อให้ผู้เขียนโปรแกรมเขียนแบบโอโอพีได้

นอกจากนี้ยังมีภาษาที่สนับสนุนแนวความคิดทางด้านโอโอพีอีกหลายภาษา เช่น Lisp, Logo, Simula, The Carpenter's Wrench เป็นต้น

บทความนี้ เป็นเพียงบทความแนะนำหลักการของโอโอพีเพื่อให้ผู้อ่านได้รู้จัก สำหรับรายละเอียดเพิ่มเติมอาจหาอ่านได้จากหนังสือทางด้านโอโอพีโดยตรง หรือจะอ่านจากคู่มือ SmallTalk ซึ่งปัจจุบันมีทั้งซอฟต์แวร์ให้ใช้กันแล้ว

เอกสารอ้างอิง

1. Timothy Budd "A Little Smalltalk" Addison-Wesley Publishiry Company, 1987.

2. Byte Special Issue on Object-Oriented Language, Aug, 1986 Vol.11, No.6